**JWT:**

- cryptographically signed JSON data

- consists of header, payload, signature (separated by a dot)

**Test:**

- modify the JWT to check if the signature is verified

- try ‘none’ Signing Algorithm

- brute-force the signing key with hashcat + well-known secrets

https://github.com/wallarm/jwt-secrets/blob/master/jwt.secrets.list

hashcat -a 0 -m 16500 <jwt> <wordlist>

+ base64-encode the secret

+ create a symmetric key with this encoded secret

+ use this key to sign the JWT

- try embedded JWK (use own key in “jwk” header)

- use “jku” header pointing to exploiting server hosting own keys

{

"keys": [

{

"kty": "RSA",

"e": "AQAB",

"kid": "75d0ef47-af89-47a9-9061-7c02a610d5ab",

"n": "o-yy1wpYmffgXBxhAUJzHHocCuJolwDqql75ZWuCQ\_cb33K2vh9mk6GPM9gNN4Y\_qTVX67WhsN3JvaFYw-fhvsWQ"

},

{

"kty": "RSA",

"e": "AQAB",

"kid": "d8fDFo-fS9-faS14a9-ASf99sa-7c1Ad5abA",

"n": "fc3f-yy1wpYmffgXBxhAUJzHql79gNNQ\_cb33HocCuJolwDqmk6GPM4Y\_qTVX67WhsN3JvaFYw-dfg6DH-asAScw"

}

]

}

- if server stores keys in filesystem, make “kid” point to ../../../../../../../../../dev/null (directory traversal)

+ symmetric key is the base64-encoded nullbyte

- if server stores keys in database, “kid” can be used for SQL injection

- Algorithm confusion attack

+ find server’s public JWK key -> check /.well-known/jwks.json, /jwks.json

+ try different key formats: PEM, JWK

Convert to PEM

1) New RSA key > paste the public JWK key > save

2) right click > Copy Public Key as PEM

3) base64-encode the PEM key

4) New Symmetric Key > Generate

5) replace value of “k” with the base64-encoded PEM key

6) set “alg” to HS256

7) sign JWT with the new symmetric key

If public key is not available, find it using

docker run --rm -it portswigger/sig2n <token1> <token2>

Use the forged JWT to check which output key is correct

Continue with step 4

- no JWT in URL